Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

ENSET

ا لمدرسة العليا لأساتذة التعليم التقني المحمدية

جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

JEE : Activité Pratique
Etudiants

« Ingénierie Informatique : Big Data et Cloud Computing »
II-BDCC2

Par : Assimi DIALLO

Dans ce TP nous allons créer une application Web basée sur Spring MVC, Spring Data JPA et Spring Security qui permet de gérer des étudiants.
Chaque étudiant est défini par:
- Son id
- Son nom
- Son prénom
- Son email
- Sa date naissance
- Son genre : MASCULIN ou FEMININ
- Un attribut qui indique si il est en règle ou non

L'application doit offrir les fonctionnalités suivantes :
- Chercher des étudiants par nom
- Faire la pagination
- Supprimer des étudiants en utilisant la méthode
- Saisir et Ajouter des étudiants avec validation des formulaires
- Editer et mettre à jour des étudiants
- Créer une page template
- Sécuriser l'accès à l'application avec un système d'authentification basé sur Spring security en utilisant la stratégie UserDetails Service

-Classe Etudiant :

```java
@Entity
@Data @AllArgsConstructor @NoArgsConstructor
public class Etudiant {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotEmpty
    @Size(min = 5, max = 20)
    private String nom;
    @NotEmpty
    @Size(min = 5, max = 20)
    private String prenom;
    @NotEmpty
    private String email;
    @Temporal(TemporalType.DATE)
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date dateNaissance;
    private Genre genre;
    private boolean enregle;

}
```

-Genre :

```java
package ma.enset.ap_etud.entities;

public enum Genre {
    MASCULIN,
    FEMININ
}
```

-EtudiantRepository

```java
package ma.enset.ap_etud.repositories;

import ma.enset.ap_etud.entities.Etudiant;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;

public interface EtudiantRepository extends JpaRepository<Etudiant, Long> {
    Page<Etudiant> findByNomContains(String keyword, Pageable pageable);
}
```

-Ajout d'étudiants dans la base de données :

```java
@Bean
CommandLineRunner saveEtudiants(EtudiantRepository etudiantRepository){
    return args->{
        etudiantRepository.save(new Etudiant( id: null, nom: "Diallo", prenom: "Toto", email: "assim@mail.com",new Date(), Genre.MASCULIN, enregle: true));
        etudiantRepository.save(new Etudiant( id: null, nom: "Ba", prenom: "Fifi", email: "ba@mail.com",new Date(), Genre.FEMININ, enregle: true));
        etudiantRepository.save(new Etudiant( id: null, nom: "Barry", prenom: "Momo", email: "barry@mail.com",new Date(), Genre.MASCULIN, enregle: true));
        etudiantRepository.save(new Etudiant( id: null, nom: "Sow", prenom: "Nana", email: "sow@mail.com",new Date(), Genre.FEMININ, enregle: false));

        etudiantRepository.findAll().forEach(e->{
            System.out.println(e.getNom()+" "+e.getPrenom());
        });
    };
}
```

-Pour la sécurité :

-Classe AppUser :

```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class AppRole {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long roleId;
    @Column(unique = true)
    private String roleName;
    private String description;
}
```

-Classe AppRole

```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class AppUser {
    @Id
    private String userId;
    @Column(unique = true)
    private String username;
    private  String password;
    private boolean active;
    @ManyToMany(fetch = FetchType.EAGER)
    private List<AppRole> appRoles = new ArrayList<>();
}
```

-AppUserRepository

```java
package ma.enset.ap_etud.sec.repositories;


import ma.enset.ap_etud.sec.entities.AppUser;
import org.springframework.data.jpa.repository.JpaRepository;

public interface AppUserRepository extends JpaRepository<AppUser, String> {
    AppUser findByUsername(String username);
}
```

-AppRoleRepository

```java
package ma.enset.ap_etud.sec.repositories;


import ma.enset.ap_etud.sec.entities.AppRole;
import org.springframework.data.jpa.repository.JpaRepository;

public interface AppRoleRepository extends JpaRepository<AppRole, Long> {
    AppRole findByRoleName(String roleName);
}
```

-SecurityService

```java
package ma.enset.ap_etud.sec.service;


import ma.enset.ap_etud.sec.entities.AppRole;
import ma.enset.ap_etud.sec.entities.AppUser;

public interface SecurityService {
    AppUser saveNewUser(String username, String password, String rePassword);
    AppRole saveNewRole(String roleName, String description);
    void addRoleToUser(String username, String roleName);
    void removeRoleFromUser(String username, String roleName);
    AppUser loadUserByUserName(String username);
}
```

-SecurityServiceImpl

```java
@Service
@Slf4j
@AllArgsConstructor
@Transactional
public class SecurityServiceImpl implements SecurityService {
    private AppUserRepository appUserRepository;
    private AppRoleRepository appRoleRepository;
    private PasswordEncoder passwordEncoder;

    @Override
    public AppUser saveNewUser(String username, String password, String rePassword) {
        if(!password.equals(rePassword)) throw new RuntimeException("Password not match");
        String hashedPWD = passwordEncoder.encode(password);
        AppUser appUser = new AppUser();
        appUser.setUserId(UUID.randomUUID().toString());
        appUser.setUsername(username);
        appUser.setPassword(hashedPWD);
        appUser.setActive(true);
        AppUser savedAppUser = appUserRepository.save(appUser);
        return savedAppUser;
    }
```

```java
    @Override
    public AppRole saveNewRole(String roleName, String description) {
        AppRole appRole = appRoleRepository.findByRoleName(roleName);
        if(appRole!=null) throw  new RuntimeException("Role "+roleName+ " already exist");
        appRole = new AppRole();
        appRole.setRoleName(roleName);
        appRole.setDescription(description);
        AppRole savedAppRole = appRoleRepository.save(appRole);
        return savedAppRole;
    }

    @Override
    public void addRoleToUser(String username, String roleName) {
        AppUser appUser = appUserRepository.findByUsername(username);
        if(appUser==null) throw  new RuntimeException("User not found");
        AppRole appRole = appRoleRepository.findByRoleName(roleName);
        if(appRole==null) throw  new RuntimeException("Role not found");
        appUser.getAppRoles().add(appRole);
        //appUserRepository.save(appUser); pas necessaire
    }

    @Override
    public void removeRoleFromUser(String username, String roleName) {
        AppUser appUser = appUserRepository.findByUsername(username);
        if(appUser==null) throw  new RuntimeException("User not found");
        AppRole appRole = appRoleRepository.findByRoleName(roleName);
        if(appRole==null) throw  new RuntimeException("Role not found");
        appUser.getAppRoles().remove(appRole);
    }

    @Override
    public AppUser loadUserByUserName(String username) { return appUserRepository.findByUsername(username); }
}
```

-UserDetailsServiceImpl

```java
@Service
public class UserDetailsServiceImpl implements UserDetailsService {
    @Autowired
    private SecurityService securityService;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        AppUser appUser = securityService.loadUserByUserName(username);

        Collection<GrantedAuthority> authorities1=
                appUser.getAppRoles() List<AppRole>
                        .stream() Stream<AppRole>
                        .map(role->new SimpleGrantedAuthority(role.getRoleName())) Stream<SimpleGrantedAuthority>
                        .collect(Collectors.toList());

        User user = new User(appUser.getUsername(), appUser.getPassword(),authorities1);
        return user;
    }
}
```

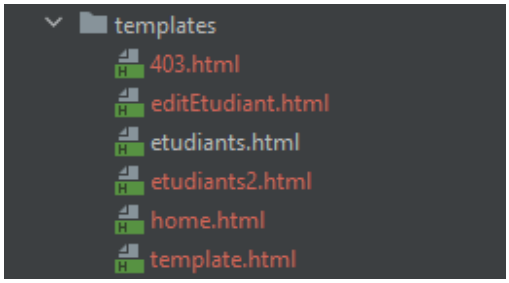
-SecurityConfig

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private UserDetailsServiceImpl userDetailsService;
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService);
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.formLogin();
        http.authorizeRequests().antMatchers( ...antPatterns: "/").permitAll();
        http.authorizeRequests().antMatchers( ...antPatterns: "/admin/**").hasAuthority("ADMIN");
        http.authorizeRequests().antMatchers( ...antPatterns: "/user/**").hasAuthority("USER");
        http.authorizeRequests().antMatchers( ...antPatterns: "/webjars/**").permitAll();
        http.authorizeRequests().anyRequest().authenticated();
        http.exceptionHandling().accessDeniedPage("/403");
    }
}
```

-Ajout des utilisateurs et des rôles :

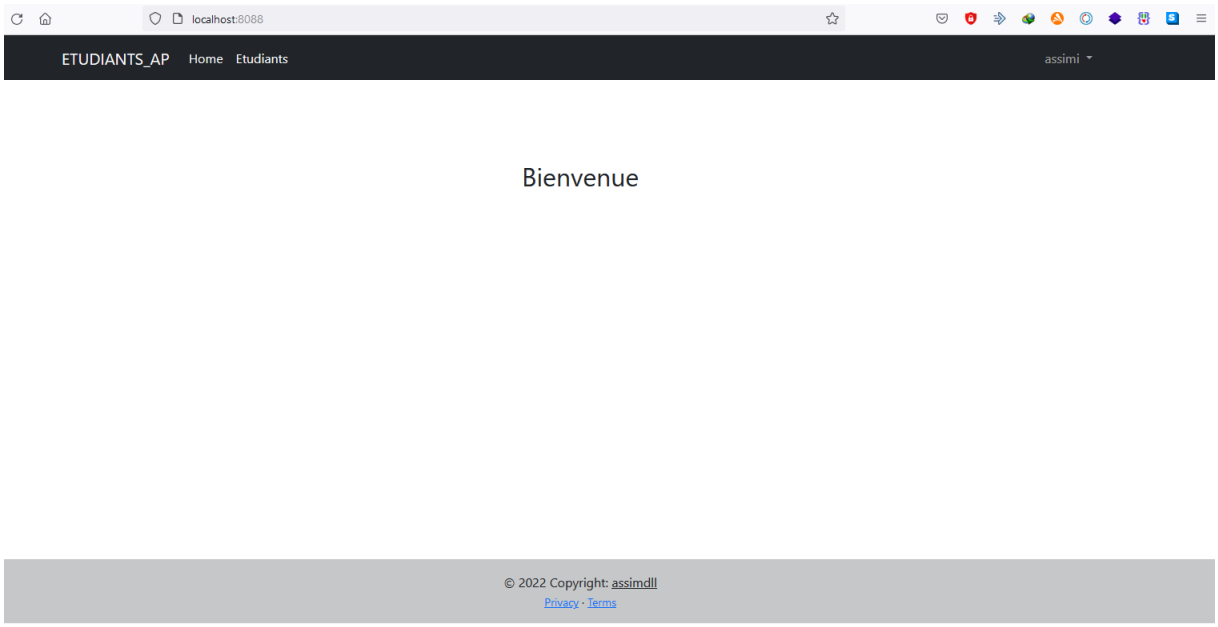
```java
@Bean
CommandLineRunner saveUsers(SecurityService securityService){
    return args->{
        securityService.saveNewUser( username: "assimi",  password: "1234",  rePassword: "1234");
        securityService.saveNewUser( username: "diallo",  password: "1234",  rePassword: "1234");
        securityService.saveNewUser( username: "assimdll",  password: "1234",  rePassword: "1234");

        securityService.saveNewRole( roleName: "USER",  description: "");
        securityService.saveNewRole( roleName: "ADMIN",  description: "");

        securityService.addRoleToUser( username: "assimi",  roleName: "USER");
        securityService.addRoleToUser( username: "assimi",  roleName: "ADMIN");
        securityService.addRoleToUser( username: "diallo",  roleName: "USER");
        securityService.addRoleToUser( username: "assimdll",  roleName: "USER");
    };
}
```

On a aussi les vues avec thymeleaf :



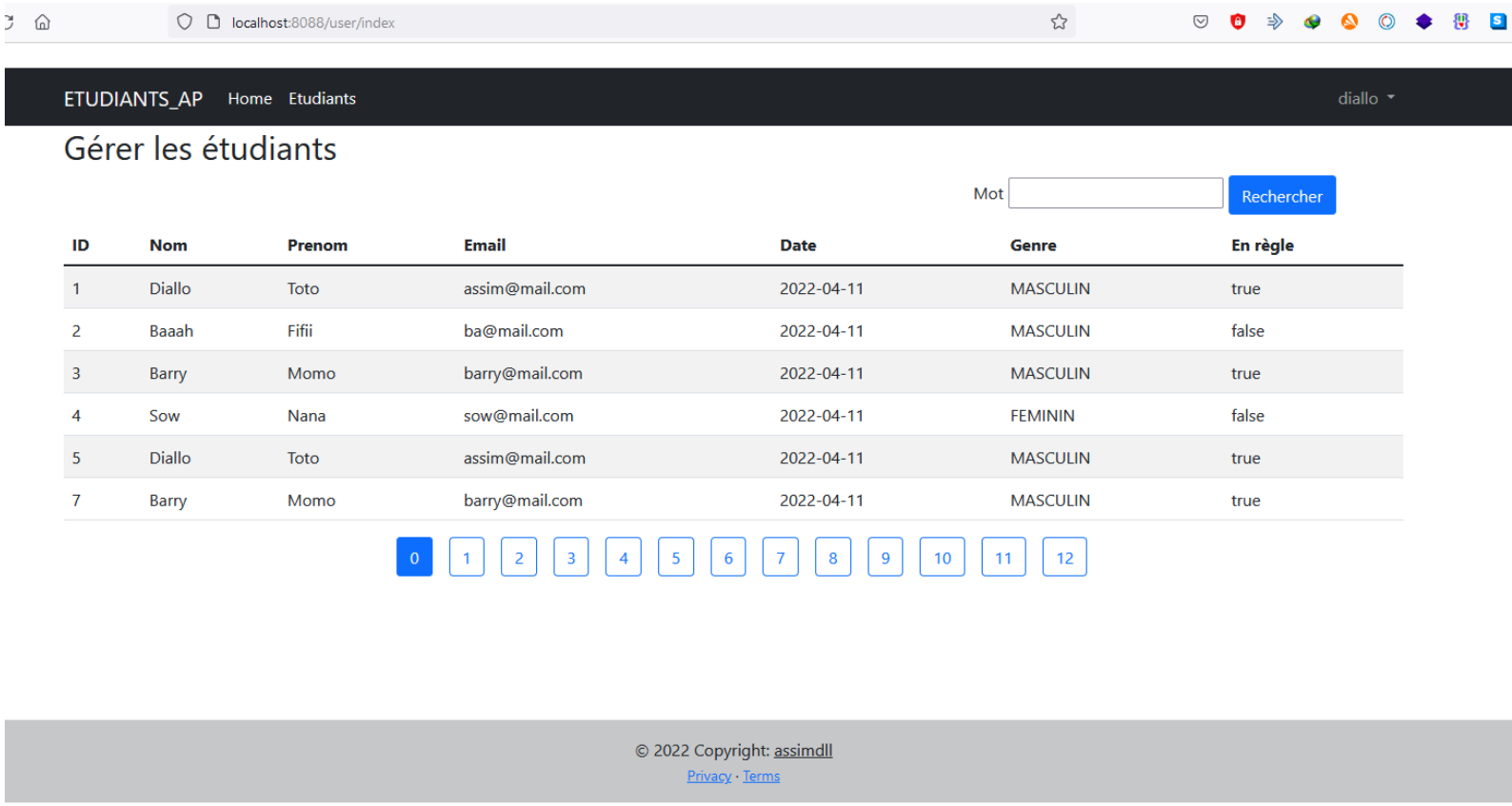
-Présentation de l'application :

-Quand on accède sur la page home :

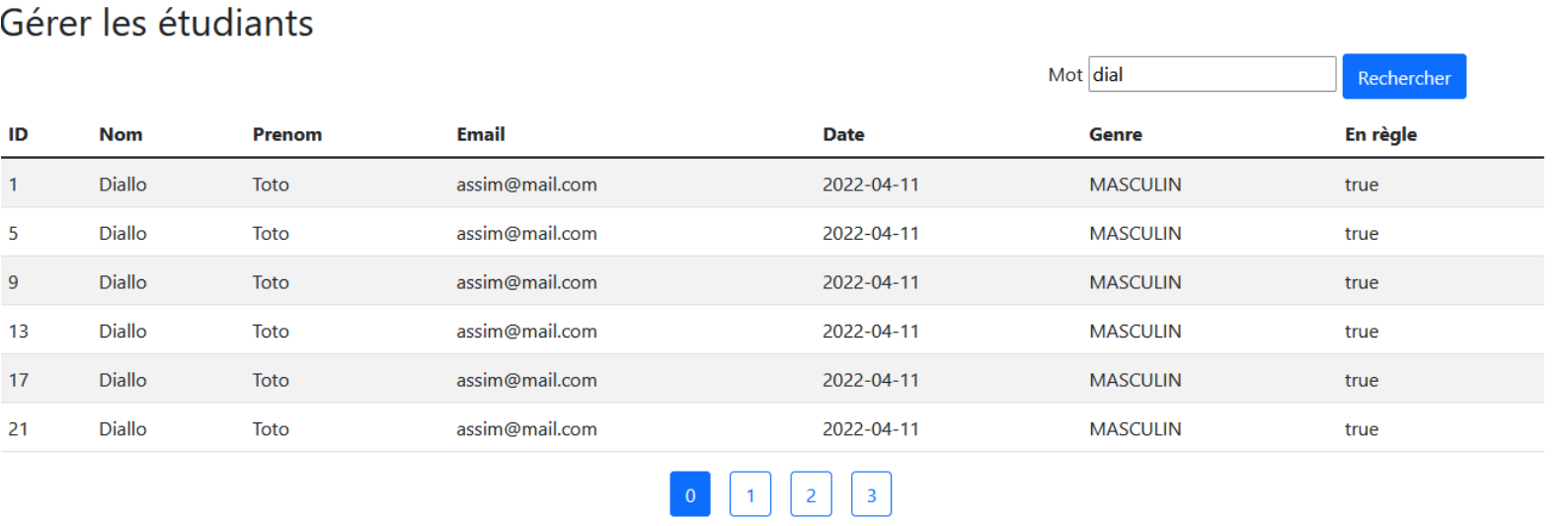


En cliquant sur **Etudiants** dans la navbar, on demande de se connecter pour accéder à la page :

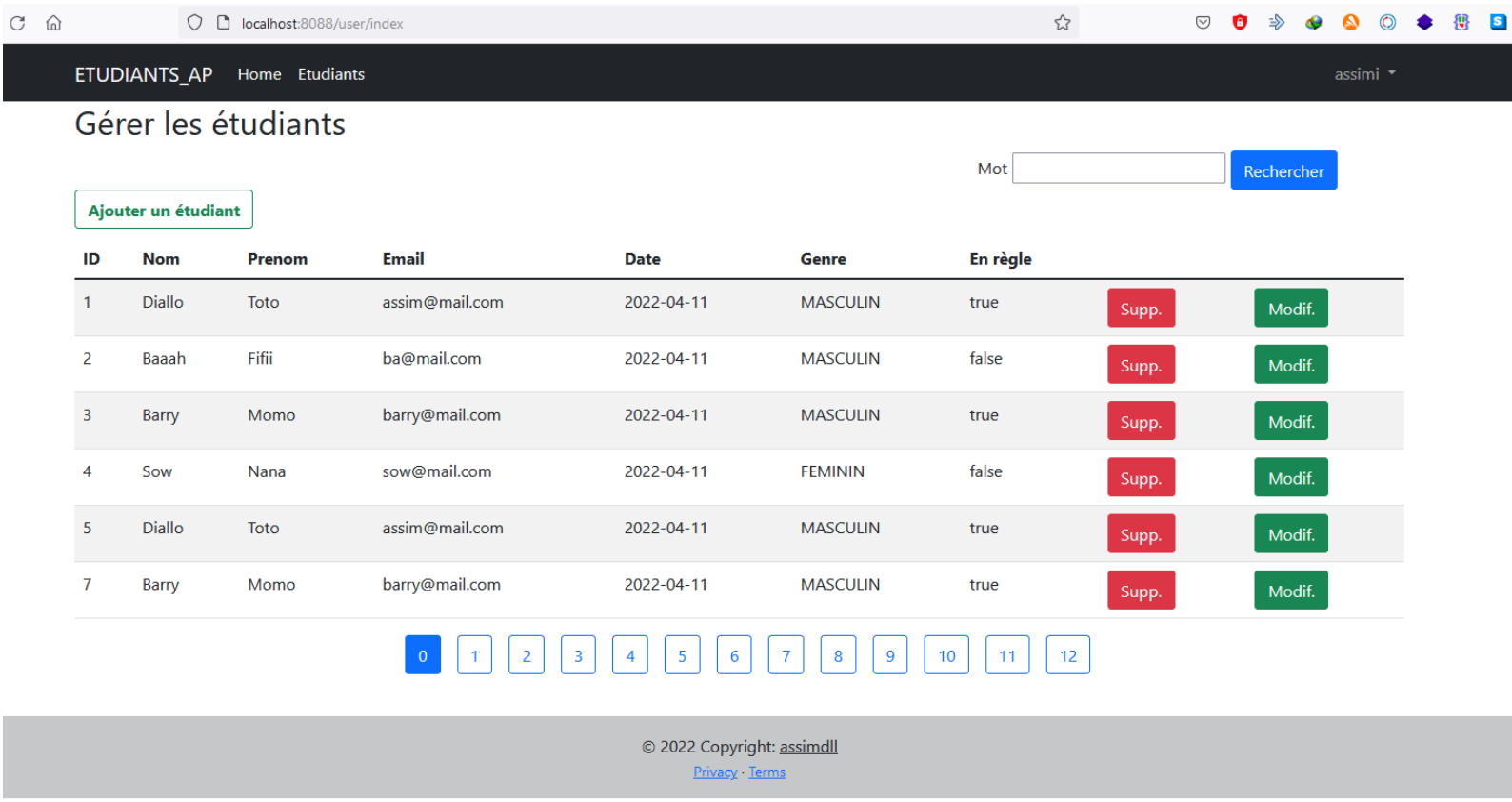
Je me connecte avec l'utilisateur **diallo** qui a un rôle USER :



Il peut voir la liste des étudiants et faire la recherche :

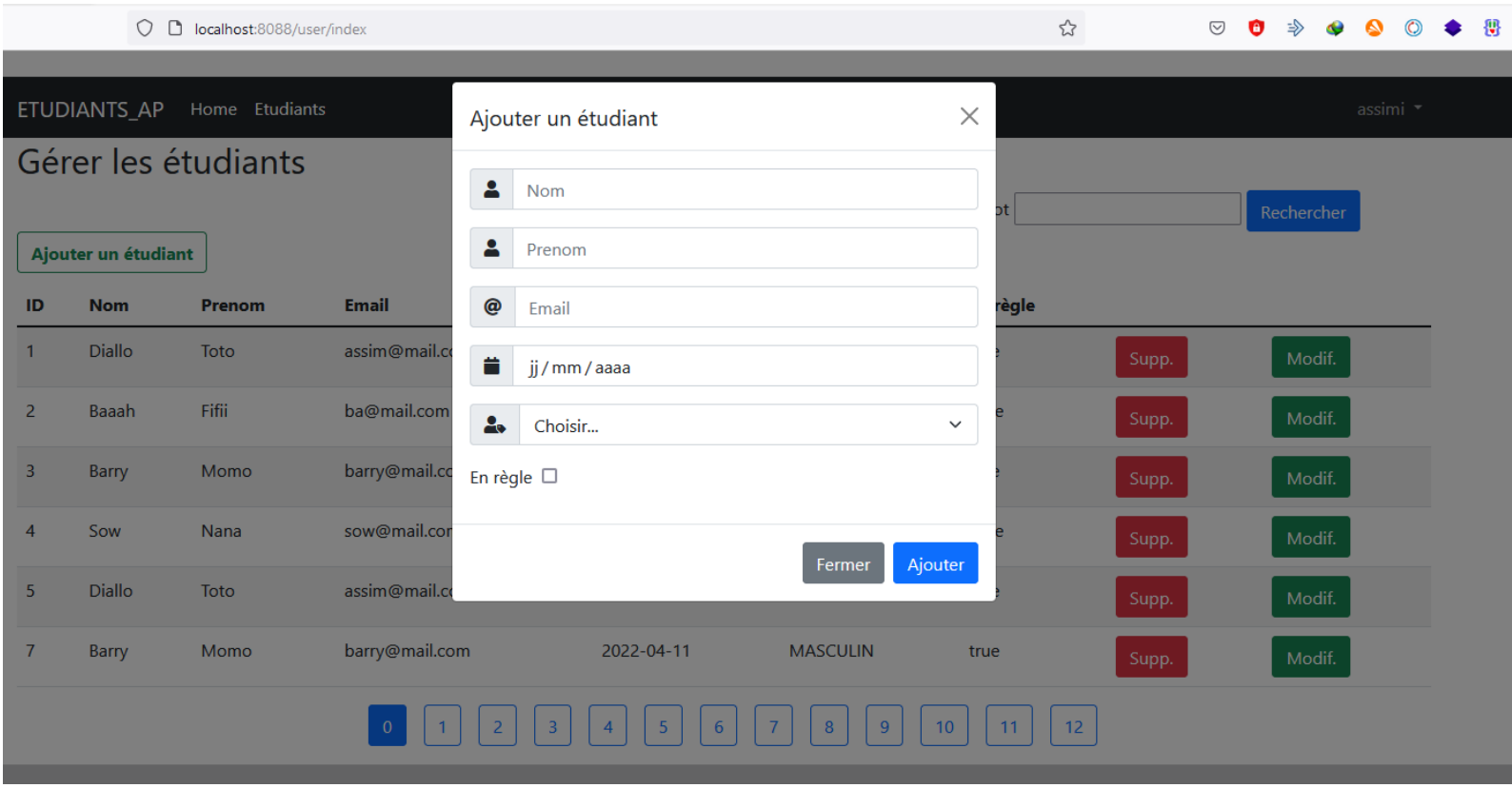


Maintenant on se déconnecte pour se connecter avec un utilisateur qui a un rôle ADMIN en plus du rôle USER.



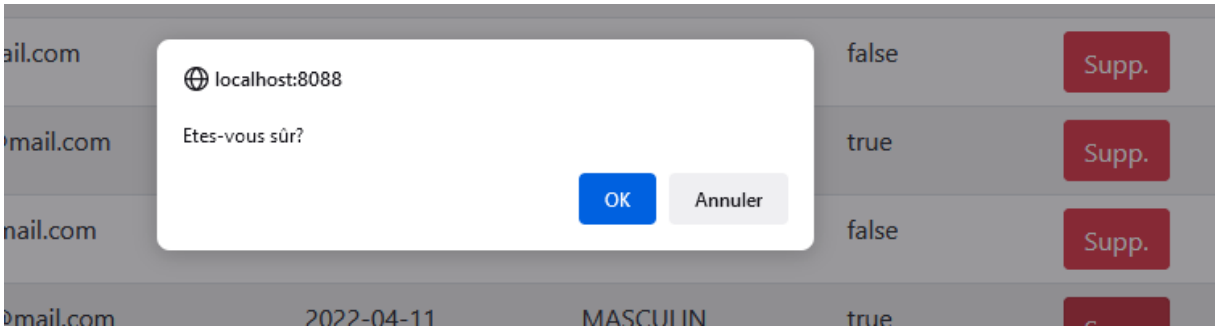
On voit donc un bouton **Ajouter un étudiant** et des boutons pour **Supprimer** et **Modifier** qui apparaissent.

Quand il clique sur le bouton **Ajouter un étudiant**

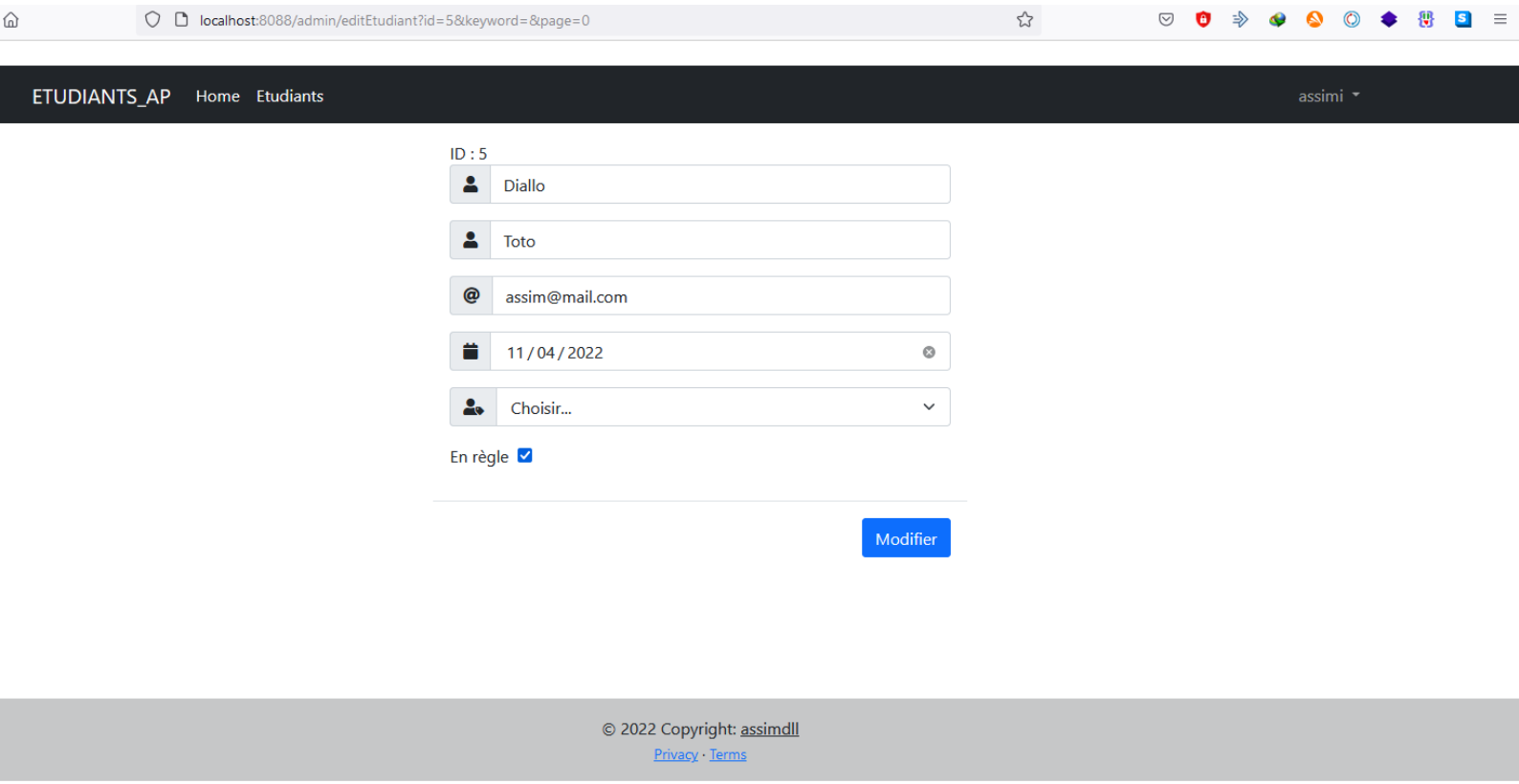


En complétant les informations et en cliquant sur le bouton **Ajouter**, l'étudiant est ajouté.

Quand il clique sur le bouton **Supp.** une confirmation pour la suppression est demandée, si on clique sur **OK**, l'étudiant est supprimé.



Quand on clique sur le bouton **Modif.** on a la possibilité de modifier l'étudiant :



Pour tout ajout ou modification d'un étudiant, il y a des règles à respecter comme le fait que le nom et le prénom doivent avoir entre 5 et 20 lettres, l'adresse mail ne pas être vide…